1.2 Removing Assignments to Dead Variables

Example:

1: \( x = y + 2; \)
2: \( y = 5; \)
3: \( x = y + 3; \)

The value of \( x \) at program points 1, 2 is over-written before it can be used.

Therefore, we call the variable \( x \) dead at these program points.
Note:

→ Assignments to dead variables can be removed
→ Such inefficiencies may originate from other transformations.
Note:

→ Assignments to dead variables can be removed
→ Such inefficiencies may originate from other transformations.

Formal Definition:

The variable \( x \) is called \textit{live at} \( u \) along the path \( \pi \) starting at \( u \) relative to a set \( X \) of variables either:

if \( x \in X \) and \( \pi \) does not contain a \textit{definition} of \( x \); or:

if \( \pi \) can be decomposed into: \( \pi = \pi_1 k \pi_2 \) such that:

\begin{itemize}
  \item \( k \) is a \textit{use} of \( x \); and
  \item \( \pi_1 \) does not contain a \textit{definition} of \( x \).
\end{itemize}
Thereby, the set of all defined or used variables at an edge $k = (_, \textit{lab}, _) \quad$ is defined by:

<table>
<thead>
<tr>
<th>lab</th>
<th>used</th>
<th>defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\text{Pos} \ (e)$</td>
<td>$\textit{Vars} \ (e)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$\text{Neg} \ (e)$</td>
<td>$\textit{Vars} \ (e)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$x = e; \ _{\text{red}}$</td>
<td>$\textit{Vars} \ (e)$</td>
<td>${x}$</td>
</tr>
<tr>
<td>$x = M[e]; \ _{\text{red}}$</td>
<td>$\textit{Vars} \ (e)$</td>
<td>${x}$</td>
</tr>
<tr>
<td>$M[e_1] = e_2; \ _{\text{red}}$</td>
<td>$\textit{Vars} \ (e_1) \cup \textit{Vars} \ (e_2)$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
A variable $x$ which is not live at $u$ along $\pi$ (relative to $X$) is called **dead** at $u$ along $\pi$ (relative to $X$).

**Example:**

\[ x = y + 2; \quad y = 5; \quad x = y + 3; \]

where $X = \emptyset$. Then we observe:

<table>
<thead>
<tr>
<th></th>
<th>live</th>
<th>dead</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>${y}$</td>
<td>${x}$</td>
</tr>
<tr>
<td>1</td>
<td>$\emptyset$</td>
<td>${x, y}$</td>
</tr>
<tr>
<td>2</td>
<td>${y}$</td>
<td>${x}$</td>
</tr>
<tr>
<td>3</td>
<td>$\emptyset$</td>
<td>${x, y}$</td>
</tr>
</tbody>
</table>
The variable \( x \) is live at \( u \) (relative to \( X \)) if \( x \) is live at \( u \) along some path to the exit (relative to \( X \)). Otherwise, \( x \) is called dead at \( u \) (relative to \( X \)).
The variable $x$ is **live** at $u$ (relative to $X$) if $x$ is live at $u$ along some path to the exit (relative to $X$). Otherwise, $x$ is called **dead** at $u$ (relative to $X$).

**Question:**

How can the sets of all dead/live variables be computed for every $u$??
The variable $x$ is live at $u$ (relative to $X$) if $x$ is live at $u$ along some path to the exit (relative to $X$). Otherwise, $x$ is called dead at $u$ (relative to $X$).

Question:

How can the sets of all dead/live variables be computed for every $u$???

Idea:

For every edge $k = (u, _, v)$, define a function $[k]^+$ which transforms the set of variables which are live at $v$ into the set of variables which are live at $u$...
Let $\mathbb{L} = 2^{\text{Vars}}$.

For $k = (\_, \text{lab}, \_)$, define $\llbracket k \rrbracket^\# = \llbracket \text{lab} \rrbracket^\#$ by:

$$
\llbracket \_ \rrbracket^\# L = L
$$

$$
\llbracket \text{Pos}(e) \rrbracket^\# L = \llbracket \text{Neg}(e) \rrbracket^\# L = L \cup \text{Vars}(e)
$$

$$
\llbracket x = e; \rrbracket^\# L = (L \setminus \{x\}) \cup \text{Vars}(e)
$$

$$
\llbracket x = M[e]; \rrbracket^\# L = (L \setminus \{x\}) \cup \text{Vars}(e)
$$

$$
\llbracket M[e_1] = e_2; \rrbracket^\# L = L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)
$$
Let $\mathbb{L} = 2^{\text{Vars}}$.

For $k = (_-, \text{lab}, _-)$, define $[k]^\# = [\text{lab}]^\#$ by:

\[
\begin{align*}
[;]^\# L &= L \\
[\text{Pos}(e)]^\# L &= [\text{Neg}(e)]^\# L = L \cup \text{Vars}(e) \\
[x = e;]^\# L &= (L \setminus \{x\}) \cup \text{Vars}(e) \\
[x = M[e];]^\# L &= (L \setminus \{x\}) \cup \text{Vars}(e) \\
[M[e_1] = e_2;]^\# L &= L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)
\end{align*}
\]

$[k]^\#$ can again be composed to the effects of $[\pi]^\#$ of paths $\pi = k_1 \ldots k_r$ by:

$[\pi]^\# = [k_1]^\# \circ \ldots \circ [k_r]^\#$
We verify that these definitions are meaningful

\[ x = y + 2; \quad y = 5; \quad x = y + 2; \quad M[y] = x; \]
We verify that these definitions are meaningful

\[ x = y + 2; \quad y = 5; \quad x = y + 2; \quad M[y] = x; \]
We verify that these definitions are meaningful.
We verify that these definitions are meaningful.
We verify that these definitions are meaningful

\[ x = y + 2; \quad y = 5; \quad x = y + 2; \quad M[y] = x; \]
We verify that these definitions are meaningful.

\[
\begin{align*}
x &= y + 2; & y &= 5; & x &= y + 2; & M[y] &= x; \\
1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 \\
\{y\} & \quad \emptyset & \quad \{y\} & \quad \{x, y\} & \quad \emptyset
\end{align*}
\]
The set of variables which are live at \( u \) then is given by:

\[
\mathcal{L}^*[u] = \bigcup \{ [[\pi]]^\# X \mid \pi : u \rightarrow^* \text{stop} \}
\]

... literally:

- The paths start in \( u \)
- As partial ordering for \( \mathbb{L} \) we use \( \subseteq = \subseteq \).
- The set of variables which are live at program exit is given by the set \( X \)
Transformation DE (Dead assignment Elimination):

\[ x = e; \]
\[ x \notin \mathcal{L}^*[v] \]
\[ x = M[e]; \]
\[ x \notin \mathcal{L}^*[v] \]
Correctness Proof:

→ Correctness of the effects of edges: If \( L \) is the set of variables which are live at the exit of the path \( \pi \), then \([\pi]^* L\) is the set of variables which are live at the beginning of \( \pi \).

→ Correctness of the transformation along a path: If the value of a variable is accessed, this variable is necessarily live. The value of dead variables thus is irrelevant.

→ Correctness of the transformation: In any execution of the transformed programs, the live variables always receive the same values as in the original program.
Computation of the sets $L^*[u]$:  

(1) Collecting constraints:

\[
\begin{align*}
L[\text{stop}] & \supseteq X \\
L[u] & \supseteq [k]^\# (L[v]) \\
k & = (u, _, v) \quad \text{edge}
\end{align*}
\]

(2) Solving the constraint system by means of RR iteration.

Since $L$ is finite, the iteration will terminate.

(3) If the exit is (formally) reachable from every program point, then the smallest solution $L$ of the constraint system equals $L^*$ since all $[k]^\#$ are distributive.
Computation of the sets $\mathcal{L}^*[u]$:

(1) Collecting constraints:

\[
\begin{align*}
\mathcal{L}[\text{stop}] & \supseteq X \\
\mathcal{L}[u] & \supseteq [k]^\#(\mathcal{L}[v]) \\
k &= (u, _, v) \quad \text{edge}
\end{align*}
\]

(2) Solving the constraint system by means of RR iteration. Since $\mathbb{L}$ is finite, the iteration will terminate.

(3) If the exit is (formally) reachable from every program point, then the smallest solution $\mathcal{L}$ of the constraint system equals $\mathcal{L}^*$ since all $[k]^\#$ are distributive.

Caveat: The information is propagated backwards !!!
Example:

```
x = M[I];
y = 1;
Neg(x > 1)
M[R] = y;
```

```
\mathcal{L}[0] \supseteq (\mathcal{L}[1] \setminus \{x\}) \cup \{I\}
\mathcal{L}[1] \supseteq \mathcal{L}[2] \setminus \{y\}
\mathcal{L}[2] \supseteq (\mathcal{L}[6] \cup \{x\}) \cup (\mathcal{L}[3] \cup \{x\})
\mathcal{L}[3] \supseteq (\mathcal{L}[4] \setminus \{y\}) \cup \{x, y\}
\mathcal{L}[4] \supseteq (\mathcal{L}[5] \setminus \{x\}) \cup \{x\}
\mathcal{L}[5] \supseteq \mathcal{L}[2]
\mathcal{L}[6] \supseteq \mathcal{L}[7] \cup \{y, R\}
\mathcal{L}[7] \supseteq \emptyset
```
Example:

\[
\begin{align*}
0 & \quad x = M[I]; \\
1 & \quad y = 1; \\
2 & \quad \text{Neg}(x > 1) \quad \text{Pos}(x > 1) \\
3 & \quad y = x \ast y; \\
4 & \quad x = x - 1; \\
5 & \quad \text{dito} \\
6 & \quad M[R] = y; \\
7 & \quad \text{dito}
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>\emptyset</td>
<td>\text{dito}</td>
</tr>
<tr>
<td>6</td>
<td>{y, R}</td>
<td>\text{dito}</td>
</tr>
<tr>
<td>5</td>
<td>{x, y, R}</td>
<td>\text{dito}</td>
</tr>
<tr>
<td>4</td>
<td>{x, y, R}</td>
<td>\text{dito}</td>
</tr>
<tr>
<td>3</td>
<td>{x, y, R}</td>
<td>\text{dito}</td>
</tr>
<tr>
<td>2</td>
<td>{x, R}</td>
<td>\text{dito}</td>
</tr>
<tr>
<td>1</td>
<td>{I, R}</td>
<td>\text{dito}</td>
</tr>
</tbody>
</table>
The left-hand side of no assignment is \textit{dead}

\textbf{Caveat:}

Removal of assignments to dead variables may kill further variables:

1
\[ x = y + 1; \]

2
\[ z = 2 \times x; \]

3
\[ M[R] = y; \]

4
\[ \emptyset \]
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:

1. \( x = y + 1; \)
2. \( z = 2 \times x; \)
3. \( y, R \)
   \[ M[R] = y; \]
4. \( \emptyset \)
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:

1. $x = y + 1$
2. $x, y, R$
3. $z = 2 \times x$
4. $y, R$
5. $M[R] = y$
6. $\emptyset$
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:

```
1  y, R
   x = y + 1;
2  x, y, R
   z = 2 * x;
3  y, R
   M[R] = y;
4  ∅
```
The left-hand side of no assignment is dead

Caveat:

Removal of assignments to dead variables may kill further variables:

1. \( y, R \)
2. \( x = y + 1; \)
3. \( z = 2 \times x; \)
4. \( M[R] = y; \)
5. \( \emptyset \)

1. \( x = y + 1; \)
2. \( \emptyset \)
3. \( M[R] = y; \)
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:
The left-hand side of no assignment is **dead**

**Caveat:**

Removal of assignments to dead variables may kill further variables:
Re-analyzing the program is inconvenient

**Idea:** Analyze **true** liveness!

$x$ is called **truly live** at $u$ along a path $\pi$ (relative to $X$), either if $x \in X$, $\pi$ does not contain a definition of $x$; or if $\pi$ can be decomposed into $\pi = \pi_1 \ k \ \pi_2$ such that:

- $k$ is a **true** use of $x$;
- $\pi_1$ does not contain any **definition** of $x$. 

236
The set of truely used variables at an edge $k = (\_ , lab , v)$ is defined as:

<table>
<thead>
<tr>
<th>lab</th>
<th>truely used</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Pos($e$)</td>
<td>$\text{Vars}(e)$</td>
</tr>
<tr>
<td>Neg($e$)</td>
<td>$\text{Vars}(e)$</td>
</tr>
<tr>
<td>$x = e;$</td>
<td>$\text{Vars}(e)$ (*)</td>
</tr>
<tr>
<td>$x = M[e];$</td>
<td>$\text{Vars}(e)$ (*)</td>
</tr>
<tr>
<td>$M[e_1] = e_2;$</td>
<td>$\text{Vars}(e_1) \cup \text{Vars}(e_2)$</td>
</tr>
</tbody>
</table>

(*): given that $x$ is truely live at $v$
Example:

1. \( x = y + 1; \)
2. \( z = 2 \times x; \)
3. \( M[R] = y; \)
4. \( \emptyset \)
Example:

1

\[ x = y + 1; \]

2

\[ z = 2 \times x; \]

3

\[ y, R \]

\[ M[R] = y; \]

4

\[ \emptyset \]
Example:

1
   \[ x = y + 1; \]
   \[ y, R \]
   \[ z = 2 \times x; \]
   \[ y, R \]
2
   \[ M[R] = y; \]
   \[ \emptyset \]
3
   \[ 4 \]
Example:

1. \( y, R \)
   \[ x = y + 1; \]

2. \( y, R \)
   \[ z = 2 \times x; \]

3. \( y, R \)
   \[ M[R] = y; \]

4. \( \emptyset \)
Example:

```
x = y + 1;
y, R
z = 2 * x;
y, R
M[R] = y;

∅
```

```
M[R] = y;
```
The Effects of Edges:

\[
\begin{align*}
\text{[;] }^\# L & = L \\
\text{[Pos}(e)\text{]}^\# L & = \text{[Neg}(e)\text{]}^\# L = L \cup Vars(e) \\
[x = e;]^\# L & = (L \setminus \{x\}) \cup Vars(e) \\
[x = M[e];]^\# L & = (L \setminus \{x\}) \cup Vars(e) \\
[M[e_1] = e_2;]^\# L & = L \cup Vars(e_1) \cup Vars(e_2)
\end{align*}
\]
The Effects of Edges:

\[
\begin{align*}
[;] L & = L \\
[\text{Pos}(e)] L & = [\text{Neg}(e)] L = L \cup \text{Vars}(e) \\
[x = e;] L & = (L \setminus \{x\}) \cup (x \in L) \ ? \ \text{Vars}(e) : \emptyset \\
[x = M[e];] L & = (L \setminus \{x\}) \cup (x \in L) \ ? \ \text{Vars}(e) : \emptyset \\
[M[e_1] = e_2;] L & = L \cup \text{Vars}(e_1) \cup \text{Vars}(e_2)
\end{align*}
\]
Note:

- The effects of edges for truely live variables are more complicated than for live variables
- Nonetheless, they are distributive !!
Note:

- The effects of edges for truely live variables are more complicated than for live variables.
- Nonetheless, they are distributive!!

To see this, consider for $D = 2^U$, $f y = (u \in y) \cdot b : \emptyset$ We verify:

$$f (y_1 \cup y_2) = (u \in y_1 \cup y_2) \cdot b : \emptyset$$
$$= (u \in y_1 \lor u \in y_2) \cdot b : \emptyset$$
$$= (u \in y_1) \cdot b : \emptyset \lor (u \in y_2) \cdot b : \emptyset$$
$$= f y_1 \lor f y_2$$
Note:

- The effects of edges for truly live variables are more complicated than for live variables.
- Nonetheless, they are distributive !!

To see this, consider for \( \mathbb{D} = 2^U \), \( f \ y = (u \in y) \ ? \ b : \emptyset \). We verify:

\[
\begin{align*}
  f (y_1 \cup y_2) &= (u \in y_1 \cup y_2) \ ? \ b : \emptyset \\
  &= (u \in y_1 \lor u \in y_2) \ ? \ b : \emptyset \\
  &= (u \in y_1) \ ? \ b : \emptyset \cup (u \in y_2) \ ? \ b : \emptyset \\
  &= f \ y_1 \cup f \ y_2
\end{align*}
\]

\[\implies \text{the constraint system yields the MOP}\]
• True liveness detects \textbf{more} superfluous assignments than repeated liveness !!!

\[
x = x - 1;\]

;
True liveness detects more superfluous assignments than repeated liveness !!!

Liveness:
• True liveness detects more superfluous assignments than repeated liveness !!!

True Liveness:

\[
x = x - 1;
\]
1.3 Removing Superfluous Moves

Example:

\[ T = x + 1; \]
\[ y = T; \]
\[ M[R] = y; \]

This variable-variable assignment is obviously useless
1.3 Removing Superfluous Moves

Example:

1. $T = x + 1;$
2. $y = T;$
3. $M[R] = y;$
4. 

This variable-variable assignment is obviously useless.

Instead of $y$, we could also store $T$. 

252
1.3 Removing Superfluous Moves

Example:

This variable-variable assignment is obviously useless

Instead of \( y \), we could also store \( T \)
1.3 Removing Superfluous Moves

Example:

1. $T = x + 1$
2. $y = T$
3. $M[R] = y$
4. $M[R] = T$

Advantage: Now, $y$ has become dead
1.3 Removing Superfluous Moves

Example:

```
1  \[T = x + 1;\]
2                   2 \[T = x + 1;\]
3  \[y = T;\]  \rightarrow  3 \[y = T;\]  \rightarrow  3 \[y = T;\]
4  \[M[R] = y;\]                             \[M[R] = T;\]                             \[M[R] = T;\]
```

Advantage: Now, \(y\) has become dead
Idea:

For each expression, we record the variable which currently contains its value

We use: \( \forall = \textit{Expr} \rightarrow 2^{\textit{Vars}} \) ...
Idea:

For each expression, we record the variable which currently contains its value

We use: $\forall = \mathit{Expr} \rightarrow 2^{\mathit{Vars}}$ and define:

$$\text{[;]\# V} \quad = \quad V$$

$$\text{[Pos(e)]\# V} e' \quad = \quad \text{[Neg(e)]\# V} e' \quad = \quad \begin{cases} \emptyset & \text{if } e' = e \\ V e' & \text{otherwise} \end{cases}$$

...
\[ [x = c;]^\# V e' = \begin{cases} (V c) \cup \{x\} & \text{if } e' = c \\ (V e') \setminus \{x\} & \text{otherwise} \end{cases} \]

\[ [x = y;]^\# V e = \begin{cases} (V e) \cup \{x\} & \text{if } y \in V e \\ (V e) \setminus \{x\} & \text{otherwise} \end{cases} \]

\[ [x = e;]^\# V e' = \begin{cases} \{x\} & \text{if } e' = e \\ (V e') \setminus \{x\} & \text{otherwise} \end{cases} \]

\[ [x = M[c];]^\# V e' = (V e') \setminus \{x\} \]

\[ [x = M[y];]^\# V e' = (V e') \setminus \{x\} \]

\[ [x = M[e];]^\# V e' = \begin{cases} \emptyset & \text{if } e' = e \\ (V e') \setminus \{x\} & \text{otherwise} \end{cases} \]

// analogously for the diverse stores
In the Example:

\[ T = x + 1; \]
\[ y = T; \]
\[ M[R] = y; \]
In the Example:

\[ T = x + 1; \]
\[ y = T; \]
\[ M[R] = y; \]

We propagate information in forward direction

At start, \( V_0 e = \emptyset \) for all \( e \);

\( \sqsubseteq \subseteq \mathbb{V} \times \mathbb{V} \) is defined by:

\[ V_1 \sqsubseteq V_2 \text{ iff } V_1 e \sqsubseteq V_2 e \text{ for all } e \]
Observation:

The new effects of edges are **distributive**:

To show this, we consider the functions:

1. \( f^x_1 \ V \ e = (V \ e) \setminus \{x\} \)
2. \( f^{e,a}_2 \ V = V \oplus \{e \mapsto a\} \)
3. \( f^{x,y}_3 \ V \ e = (y \in V \ e) \Rightarrow (V \ e \cup \{x\}) : ((V \ e) \setminus \{x\}) \)

Obviously, we have:

\[
\begin{align*}
[x = e;]^{\#} &= f^{e,\{x\}}_2 \circ f^x_1 \\
[x = y;]^{\#} &= f^{x,y}_3 \\
[x = M[e];]^{\#} &= f^{e,\emptyset}_2 \circ f^x_1
\end{align*}
\]

By closure under **composition**, the assertion follows
(1) For $f V e = (V e) \setminus \{x\}$, we have:

$$f (V_1 \sqcup V_2) e = ((V_1 \sqcup V_2) e) \setminus \{x\}$$
$$= ((V_1 e) \cap (V_2 e)) \setminus \{x\}$$
$$= ((V_1 e) \setminus \{x\}) \cap ((V_2 e) \setminus \{x\})$$
$$= (f V_1 e) \cap (f V_2 e)$$
$$= (f V_1 \sqcup f V_2) e$$
(2) For \( f \ V = V \uplus \{e \mapsto a\} \), we have:

\[
\begin{align*}
  f \ (V_1 \uplus V_2) \ e' & = \ ((V_1 \uplus V_2) \uplus \{e \mapsto a\}) \ e' \\
 & = \ (V_1 \uplus V_2) \ e' \\
 & = \ (f \ V_1 \uplus f \ V_2) \ e' \quad \text{given that} \quad e \neq e' \\
\end{align*}
\]

\[
\begin{align*}
  f \ (V_1 \uplus V_2) \ e & = \ ((V_1 \uplus V_2) \uplus \{e \mapsto a\}) \ e \\
 & = \ a \\
 & = \ ((V_1 \uplus \{e \mapsto a\}) \ e) \cap ((V_2 \uplus \{e \mapsto a\}) \ e) \\
 & = \ (f \ V_1 \uplus f \ V_2) \ e
\end{align*}
\]
(3) For \( f V e = (y \in V e) \cup (V e \cup \{x\}) : ((V e)\setminus\{x\}) \), we have:

\[
f (V_1 \cup V_2) e &= (((V_1 \cup V_2) e)\setminus\{x\}) \cup (y \in (V_1 \cup V_2) e) \cup \{x\} : \emptyset \\
&= ((V_1 e \cap V_2 e)\setminus\{x\}) \cup (y \in (V_1 e \cap V_2 e) \cup \{x\} : \emptyset \\
&= ((V_1 e \cap V_2 e)\setminus\{x\}) \cup \\
&\quad ((y \in V_1 e) \cup \{x\} : \emptyset) \cap ((y \in V_2 e) \cup \{x\} : \emptyset) \\
&= (((V_1 e)\setminus\{x\}) \cup (y \in V_1 e) \cup \{x\} : \emptyset) \cap \\
&\quad (((V_2 e)\setminus\{x\}) \cup (y \in V_2 e) \cup \{x\} : \emptyset) \\
&= (f V_1 \cup f V_2) e
\]
We conclude:

→ Solving the constraint system returns the MOP solution
→ Let $\mathcal{V}$ denote this solution.

If $x \in \mathcal{V}[u] e$, then $x$ at $u$ contains the value of $e$ — which we have stored in $T_e$

$\implies$ the access to $x$ can be replaced by the access to $T_e$

For $V \in \mathcal{V}$, let $V^-$ denote the variable substitution with:

$$V^- x = \begin{cases} T_e & \text{if } x \in V e \\ x & \text{otherwise} \end{cases}$$

if $Ve \cap Ve' = \emptyset$ for $e \neq e'$. Otherwise: $V^- x = x$
Transformation CE:

\[ u \xrightarrow{\text{Pos}(e)} u \]

\[ \sigma = V[u]^- \]

\[ u \xrightarrow{\text{Pos}(\sigma(e))} u \]

... analogously for edges with \( \text{Neg}(e) \)

\[ u \xrightarrow{x = e;} u \]

\[ \sigma = V[u]^- \]

\[ u \xrightarrow{x = \sigma(e);} u \]
Transformation CE  (cont.):

\[ x = M[e]; \quad \sigma = \mathcal{V}[u]^- \]

\[ x = M[\sigma(e)]; \]

\[ M[e_1] = e_2; \quad \sigma = \mathcal{V}[u]^- \]

\[ M[\sigma(e_1)] = \sigma(e_2); \]
Procedure as a whole:

(1) Availability of expressions:
   + removes arithmetic operations
   - inserts superfluous moves

(2) Values of variables:
   + creates dead variables

(3) (true) liveness of variables:
   + removes assignments to dead variables
Example: \( a[7]--; \)

\[
\begin{align*}
A_1 &= A + 7; \\
B_1 &= M[A_1]; \\
B_2 &= B_1 - 1; \\
A_2 &= A + 7; \\
M[A_2] &= B_2;
\end{align*}
\]

\[
\begin{align*}
T_1 &= A + 7; \\
A_1 &= T_1; \\
B_1 &= M[A_1]; \\
T_2 &= B_1 - 1; \\
B_2 &= T_2; \\
T_1 &= A + 7; \\
A_2 &= T_1; \\
M[A_2] &= B_2;
\end{align*}
\]
Example: $a[7]--;$

$A_1 = A + 7;$

$B_1 = M[A_1];$

$B_2 = B_1 - 1;$

$A_2 = A + 7;$

$M[A_2] = B_2;$

$T_1 = A + 7;$

$A_1 = T_1;$

$B_1 = M[A_1];$

$T_2 = B_1 - 1;$

$B_2 = T_2;$

$T_1 = A + 7;$

$A_2 = T_1;$

$M[A_2] = B_2;$

$T_1 = A + 7;$

$A_1 = T_1;$

$B_1 = M[A_1];$

$T_2 = B_1 - 1;$

$B_2 = T_2;$

$;$

$A_2 = T_1;$

$M[A_2] = B_2;$
Example (cont.): \[ a[7] \] --;

\[
T_1 = A + 7;
\]

\[
A_1 = T_1;
\]

\[
B_1 = M[A_1];
\]

\[
T_2 = B_1 - 1;
\]

\[
B_2 = T_2;
\]

\;

\[
A_2 = T_1;
\]

\[
M[A_2] = B_2;
\]

\[
T_1 = A + 7;
\]

\[
A_1 = T_1;
\]

\[
B_1 = M[T_1];
\]

\[
T_2 = B_1 - 1;
\]

\[
B_2 = T_2;
\]

\;

\[
A_2 = T_1;
\]

\[
M[T_1] = T_2;
\]
Example (cont.): \[ a[7] -- ; \]

\[
\begin{align*}
T_1 &= A + 7; \\
A_1 &= T_1; \\
B_1 &= M[A_1]; \\
T_2 &= B_1 - 1; \\
B_2 &= T_2; \\
\vdots \\
A_2 &= T_1; \\
M[A_2] &= B_2;
\end{align*}
\]

\[
\begin{align*}
T_1 &= A + 7; \\
A_1 &= T_1; \\
B_1 &= M[T_1]; \\
T_2 &= B_1 - 1; \\
B_2 &= T_2; \\
\vdots \\
A_2 &= T_1; \\
M[T_1] &= T_2;
\end{align*}
\]

\[
\begin{align*}
T_1 &= A + 7; \\
A_1 &= T_1; \\
B_1 &= M[T_1]; \\
T_2 &= B_1 - 1; \\
B_2 &= T_2; \\
\vdots \\
A_2 &= T_1; \\
M[T_1] &= T_2;
\end{align*}
\]